## REMARKS

Claims 4-5 and 7-20 have been canceled above, and claims 21-35 have been added.

Original claim 1 was rejected under 35 USC 102 based on Sonkin et al (US Patent 7,136,868). Original claims 2-3 were rejected under 35 USC 103 based on Sonkin et al. and Chandra et al. (US Patent 6,058,389). Original claim 6 was rejected under 35 USC 103 based on Sonkin et al., Chandra et al. and Clark et al. (US Patent 7,111,063).

Amended claim 1 recites a computer program product for resetting a first application or application instance. First program instructions, responsive to a request to reset the first application or application instance, identify the objects associated with the first application or application instance. Second program instructions, responsive to the request to reset the first application or application instance, determine one or more of the objects associated with the first application or application instance which were defined by a user of the first application or application instance and determine one or more of the objects associated with the first application or application instance which were automatically created by a computer system including the first application or application instance. Third program instructions, responsive to the request to reset the first application or application instance, write a script program to command the first application or application instance to delete the one or more objects, associated with the first application or application instance, which were created by the user of the first application or application instance. The computer program product does not, in response to the request to reset the first application or application instance, command deletion of the one or more objects, associated with the first application or application instance, which were automatically created by the computer system, such that the one or more objects, associated with the first application or application instance, which were automatically created by the computer system are not deleted in response to the request to reset the first application or application instance.

Sonkin et al. discloses a technique to determine a dependency tree of objects:

"An exemplary phase or module ... creates a hierarchical object tree from one or more object references that are passed in. The complex objects, which use object references, may be expressed with uniform resource names. the module eliminates duplicate object references and produces a dependency tree. the module also offers the opportunity to edit the tree as it is being created as well as after creation. Another exemplary module or phase inputs a hierarchical dependency tree, either from a user or from the previous module and produces a dependency list. The dependency list is a linear list expressing the order of creation that an object may use in order to satisfy dependency constraints. This module also offers the opportunity to edit the dependency list as it is being created as well as after completion. Another exemplary module or phase generates script from a dependency list. The dependency list may be either user generated or it may be input from a previous module. This module instantiates the object on the dependency list and calls a scripting method corresponding to the object. This module offers a wide range of flexibility for the user or controlling program to edit the script as it is being generated as well as after completion." Sonkin et al. Column 2 lines 3-25

Thus, Sonkin et al. is concerned with determining a dependency tree, and does not disclose or even suggest a technique for resetting an application or application instance as recited in amended claim 1. For example, Sonkin et al. does not disclose or even suggest deleting objects used by an application or application instance. Moreover, Sonkin et al does not disclose or even suggest different treatment of computer system-defined objects vs. user-defined objects. According to amended claim 1, the user-defined objects are deleted but the computer system-defined objects are not deleted, in the resetting process. The computer system-defined objects survive the reset. Also, according to amended claim 1, the program instructions write a script program to command the first application or application instance to delete the one or more user-defined objects. In contrast, the script of Sonkin et al. "permits the deployment of varying complexity objects onto a target database". Sonkin et al. Abstract. "The dependency list may then be processed into a script that can be used to deploy relational database objects in a given target database." Sonkin et al. Column 2 lines 63-66. Therefore, Sonkin et al. is far afield of the present invention and has little to do with a technique for resetting an application or application

instance by automatically identifying and deleting certain types of objects associated with the application or application instance as recited in amended claim 1.

Claims 2-4 and 6 depend on amended claim 1 and therefore, distinguish over the cited prior art for the same reasons that amended claim 1 distinguishes thereover.

Dependent claim 4 further distinguishes over Sonkin et al. because claim 4 recites a technique for removing a lock held by a remote application on a message queue object, i.e. stopping a channel connection of the remote application to the queue object, before attempting to delete the queue object or its work items. Sonkin et al. is not concerned with queue objects or their channel connections, but instead is concerned with relational database objects. See Sonkin et al. Column 1 lines 66-67 and Column 2 lines 56-66. A message queue object is fundamentally different than a relational database object, because a relational database stores indexed data whereas a message queue is a temporarily staging area to forward messages to their intended destination. Chandra et al. does not fill this gap of Sonkin et al. Chandra et al. disclose:

> "the function of the DROP_Q process is to delete or drop a specific, uniquely named queue from an existing queue table and from storage in the system. The DROP_Q process accepts two parameters, Queue Name and Auto-Commit. The Queue Name parameter identifies the queue to be dropped. The Auto-Commit parameter accepts a Boolean value; a TRUE value specifies that the current transaction, if any, must commit before the drop operation is carried out." Chandra et al. Column 33 lines 34-44.

However, in contrast to present claim 4, Chandra et al. does not disclose deletion of a message queue as part of an automated process of resetting an application or application instance. Also, Chandra et al. does not disclose stopping of a channel connection of a remote application to a queue object to remove a lock on the queue object to enable a first application or application instance to delete the queue object. Also, Chandra et al. does not disclose a computer program product to automatically generate a script to command the first application or application instance to delete a message queue as part of a resetting process. Bhattal et al. does not fill the

gap of Sonkin et al. and Chandra et al. either. Bhattal et al. discloses a technique to recover from a failure including failure of a channel:

> "If a queue manager fails (scenario 4) this will necessarily mean that the paired channel initiator will also fail. This triggers the failure event being logged at all currently active queue sharing group queue managers. Each of these queue managers (Y and Z) then enters similar processing to that described above, to implement peer recovery. When there are multiple queue managers performing recovery, an attempt to take ownership of a channel issued by a certain queue manager may fail because another queue manager has already processed the entry and changed the owning queue manager name. Only one start request is actioned for a channel, but different queue managers may recover different channels." Bhattal et al. Column 9 lines 28-40.

However, in contrast to claim 4, Bhattal et al. does not disclose deletion of a queue object as part of a resetting process. Also, Bhattal et al. does not disclose stopping of a channel connection of a remote application to a queue object to remove a lock on the queue object to enable a first application or application instance to delete the queue object. Also, Chandra et al. does not disclose a computer program product to automatically generate a script to command the first application or application instance to delete a message queue as part of a resetting process.

New claim 21 distinguishes over the cited prior art for the same reasons that amended claim 1 distinguishes thereover.

New claims 22-23 depend on new claim 21 and therefore, distinguish over the cited prior art for the same reasons that new claim 21 distinguishes thereover.

New claim 23 further distinguishes over the prior art for the same reasons that new claim 4 further distinguishes over the prior art.

New independent claim 24 recites a computer program product for resetting a first application or application instance. First program instructions, responsive to a request to reset

the first application or application instance, identify for deletion a queue object used or managed by the first application or application instance. Second program instructions, responsive to the request to reset the first application or application instance, identify a channel used by another, remote application or application instance to lock and access the queue object, and write into a script program a command to the first application or application instance to stop the channel, wherein a lock held by the remote application or application instance on the queue object will automatically be released in response to execution of the command to stop the channel. Third program instructions, responsive to the request to reset the first application or application instance, write into the script program another command, for execution after the command to stop the channel, to the first application or application instance to delete work items on the queue object and/or delete the queue object.

As explained above, Sonkin et al. discloses a technique to determine a dependency tree of objects. Thus, Sonkin et al. does not disclose or even suggest a technique for resetting an application or application instance as recited in claim 24. For example, Sonkin et al. do not disclose or even suggest deleting an object used by an application. Moreover, Sonkin et al do not disclose or even suggest writing a script program to command the first application or application instance to delete the queue object. In contrast, the script of Sonkin et al. "permits the deployment of varying complexity objects onto a target database". Sonkin et al. Abstract. "The dependency list may then be processed into a script that can be used to deploy relational database objects in a given target database." Sonkin et al. Column 2 lines 63-66. Therefore, Sonkin et al. is far afield of the present invention and has little to do with a technique for resetting an application or application instance by automatically writing a script to delete a queue object as recited in new claim 24.

New independent claim 24 further distinguishes over Sonkin et al. because claim 24 recites a technique for removing a lock held by a remote application on a queue object, i.e. stopping a channel connection of the remote application to the queue object, before attempting to delete the queue object or work items on the queue object. Sonkin et al. is not concerned with queue objects or their channel connections, but instead is concerned with relational database objects. See Sonkin et al. Column 1 lines 66-67 and Column 2 lines 56-66. Aa message queue

object is fundamentally different than a relational database object, because a relational database stores indexed data whereas a message queue is a temporarily staging area to forward messages to their intended destination. Chandra et al. does not fill this gap of Sonkin et al. Chandra et al. discloses:

> "the function of the DROP_Q process is to delete or drop a specific, uniquely named queue from an existing queue table and from storage in the system. The DROP_Q process accepts two parameters, Queue Name and Auto-Commit. The Queue Name parameter identifies the queue to be dropped. The Auto-Commit parameter accepts a Boolean value; a TRUE value specifies that the current transaction, if any, must commit before the drop operation is carried out." Chandra et al. Column 33 lines 34-44.

However, in contrast to new claim 24, Chandra et al. does not disclose deletion of a message queue as part of an automated process of resetting an application. Also, Chandra et al. does not disclose stopping of a channel connection of a remote application to a queue object to remove a lock on the queue object to enable a first application or application instance to delete the queue object or work items on the queue object. Also, Chandra et al. does not disclose a computer program product to automatically generate a script to command the first application or application instance to delete a message queue as part of a resetting process. Bhattal et al. does not fill the gap of Sonkin et al. and Chandra et al. either. Bhattal et al. discloses a technique to recover from a failure including failure of a channel:

> "If a queue manager fails (scenario 4) this will necessarily mean that the paired channel initiator will also fail. This triggers the failure event being logged at all currently active queue sharing group queue managers. Each of these queue managers (Y and Z) then enters similar processing to that described above, to implement peer recovery. When there are multiple queue managers performing recovery, an attempt to take ownership of a channel issued by a certain queue manager may fail because another queue manager has already processed the entry and changed the owning queue manager name. Only one start request is actioned for a channel, but different queue managers may recover different channels." Bhattal et al. Column 9 lines 28-40.

However, in contrast to claim 24, Bhattal et al. does not disclose deletion of a queue object as part of a resetting process. Also, Bhattal et al. does not disclose stopping of a channel connection of a remote application to a queue object to remove a lock on the queue object to enable a first application or application instance to delete the queue object. Also, Chandra et al. does not disclose a computer program product to automatically generate a script to command the first application or appliciation instance to delete a message queue as part of a resetting process.

Claims 25-29 depend on new claim 24 and therefore, distinguish over the cited prior art for the same reasons that new claim 24 distinguishes thereover.

New claim 30 distinguishes over the cited prior art for the same reasons that new claim 24 distinguishes thereover.

New claims 31-35 depend on new claim 30 and therefore, distinguish over the cited prior art for the same reasons that new claim 24 distinguishes thereover.

Based on the foregoing, the present patent application as amended above should be allowed.

Respectfully submitted,

Dated: 04/10/2007

/Arthur J. Samodovitz/

Telephone: 607-429-4368
Fax No.: 607-429-4119

Arthur J. Samodovitz
Reg. No. 31,297